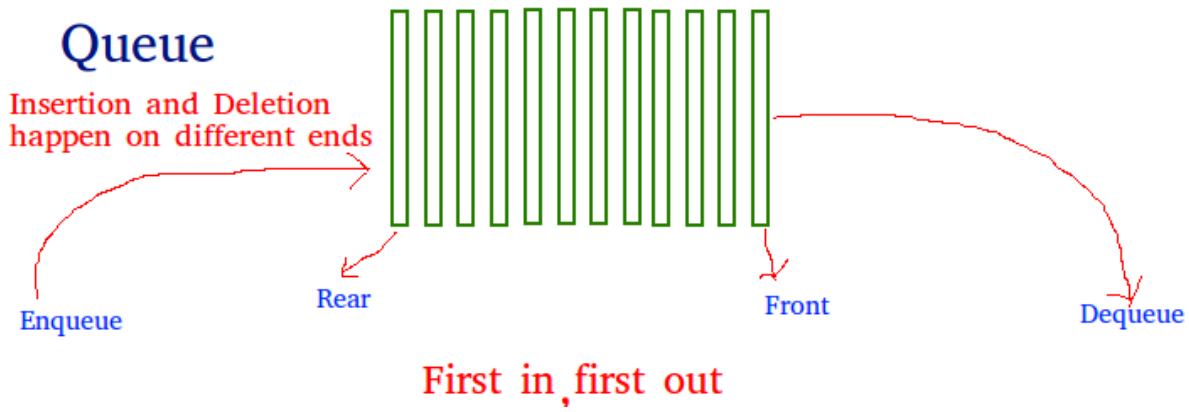
Date=10/08/2020
Lecture By=Shubham Joshi
Subject ⇒Queue and Tree

IN PREVIOUS LECTURE **(QUICK RECAP) Date-06/08/2020**	**In Today's Lecture (Overview)**
Questions Regarding Stacks Question=Trapping Rain Water MCQ's Questions For Self Practice // CC AND Assignment For The Day	Queue In Python Front And Rear in Queue In Python Python Program To Implement Queue In Python Question Regarding Queue Trees In Python What is Leaf Node Complete Binary Tree Full Binary Tree MCQs Questions For Self Practice

Queue In Python

Like stack, queue is a linear data structure that stores items in First In First Out (FIFO) manner
With a queue the least recently added item is removed first.
A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first.



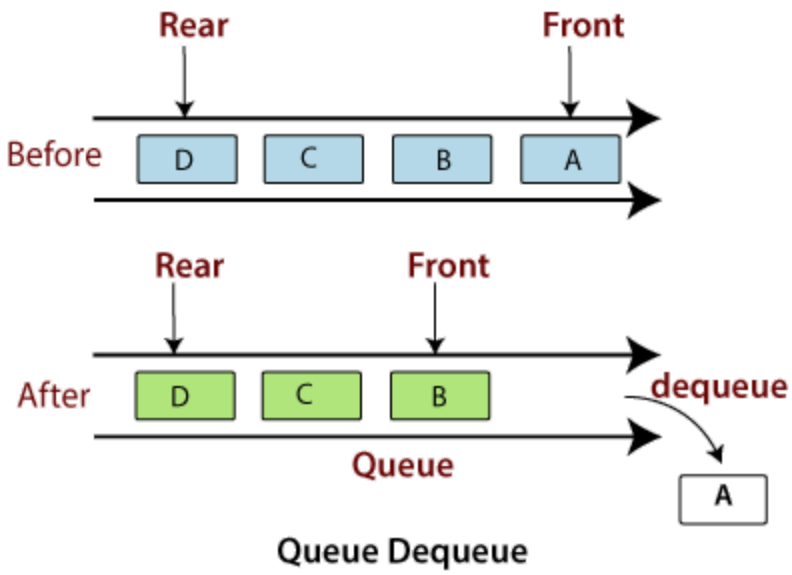
Operations associated with queue are:

- **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition – Time Complexity : O(1)
- **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition – Time Complexity : O(1)
- **Front:** Get the front item from queue – Time Complexity : O(1)
- **Rear:** Get the last item from queue – Time Complexity : O(1)

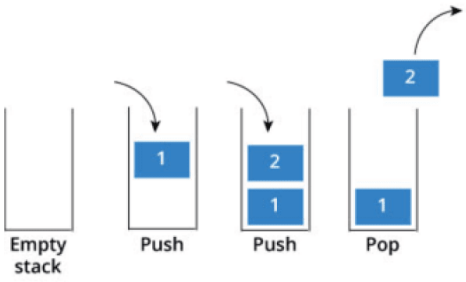
Front And Rear in Queue In Python

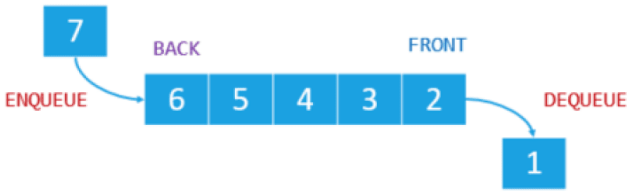
Front
-Element To be Removed From the queue
-Basically its the first element of the queue

Rear
-Element To be Inserted From The Queue
-Basically its the last element of the queue



Data Structure Basics





Python Program To Implement Queue In Python

```
# Python program to
# demonstrate queue implementation
# using list

# Initializing a queue
```

```
queue = []

# Adding elements to the queue
queue.append('a')
queue.append('b')
queue.append('c')

print("Initial queue")
print(queue)

# Removing elements from the queue
print("\nElements dequeued from queue")
print(queue.pop(0))
print(queue.pop(0))
print(queue.pop(0))

print("\nQueue after removing elements")
print(queue)

# Uncommenting print(queue.pop(0))
# will raise and IndexError
# as the queue is now empty
```

Output:

```
Initial queue
['a', 'b', 'c']

Elements dequeued from queue
a
b
c

Queue after removing elements
[]
```

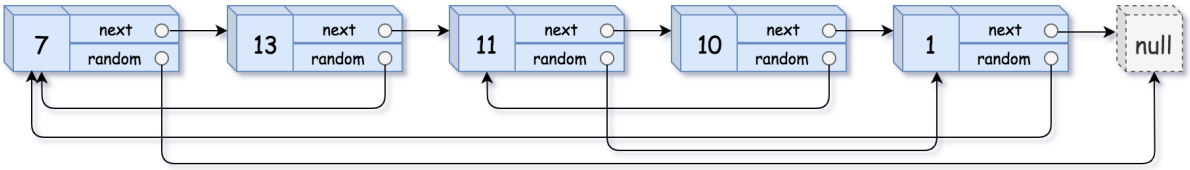
Question Regarding Queue

Q1.Copy List With Random Pointers(Leetcode)

A linked list is given such that each node contains an additional random pointer which could point to any node in the list or null.

Return a deep copy of the list.

The Linked List is represented in the input/output as a list of n nodes. Each node is represented as a pair of [val, random_index] where
Example



Input: head = [[7,null],[13,0],[11,4],[10,2],[1,0]]
Output: [[7,null],[13,0],[11,4],[10,2],[1,0]]

Code

```
"""
class Node(object):
    def __init__(self, val, next, random):
        self.val = val
        self.next = next
        self.random = random
"""
class Solution(object):
    def copyRandomList(self, head):
        """
        :type head: Node
        :rtype: Node
        """
        deep_copy_dict={}
        if not head:
            return None
```

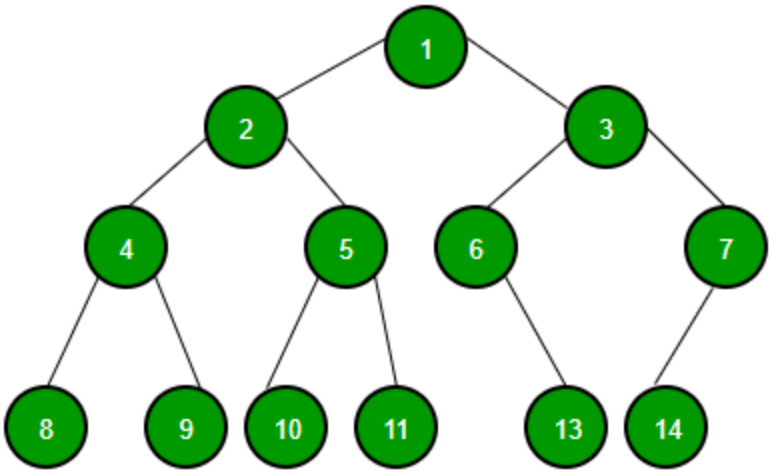
```
        current_node=head
        while current_node:
            deep_copy_dict[current_node]=Node(current_node.val,None,None)
            current_node=current_node.next
        current_node=head
        while current_node:
            if current_node.next:

deep_copy_dict[current_node].next=deep_copy_dict.get(current_node.next)
            if current_node.random:

deep_copy_dict[current_node].random=deep_copy_dict.get(current_node.random
)
            current_node=current_node.next

        return deep_copy_dict.get(head)
```

This Question Link https://leetcode.com/problems/copy-list-with-random-pointer/

Trees In Python

Tree represents the nodes connected by edges.
It is a non-linear data structure. It has the following properties.

- One node is marked as Root node.
- Every node other than the root is associated with one parent node.
- Each node can have an arbitrary number of child node.



What is Leaf Node

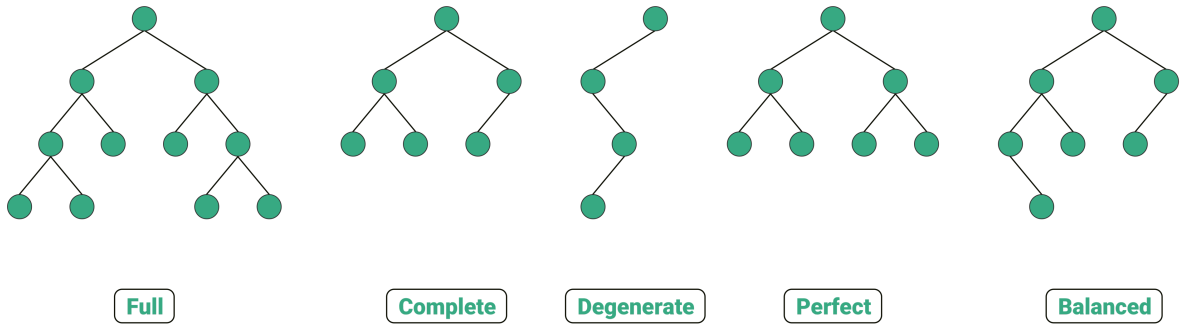
=Leaf Node is A Node which Does Not Any Children

Complete Binary Tree

A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

Full Binary Tree

A **full binary tree** (sometimes proper **binary tree** or 2-**tree**) is a **tree** in which every node other than the leaves has two children



MCQs

1.How many nodes will a binary tree of level 4 have?

A.15

B.7

C.5

D.1

2.What is the time complexity to pop an element from queue in its linked list implementation ?

A.O(n)

B.O(1)

C.O(n2)

3.**which of these is a property of queue**

A.LIFO

B.FIFO

C.we can access any element in a queue.

4.what will be the output of following in a queue ? queue.push(5) q.push(7) queue.pop() queue.pop()

A.5 5

B.7 5

C.5 7

Questions For Self Practice

1.Implement Queue using linked list

2.https://leetcode.com/problems/copy-list-with-random-pointer/